HoGent

FRONT-END ANGULAR

Graduaatsproef

Shakira Hamers
6-5-2023

ASSOCIATE DEGREE AFSTUDEERWERKSTUK

FRONT-END ANGULAR

Door

Shakira Hamers

187365sh

Graduaatsproef

Graduaat Programmeren

Hogeschool Gent

Onder begeleiding van

Lector graduaatsproef

Luc Vervoort

Inhoudstafel

Samenvatting

Als opdracht voor onze graduaatsproef kregen wij de mogelijkheid om zelf een nieuwe programmeertaal aan te leren en deze toe te passen (front-end of backend) op een groepswerk dat we gemaakt hebben voor een bedrijf. In ons geval moesten wij voor een Gents IT consultancybedrijf AllPhi een bezoekersregistratiesysteem opbouwen met C# in een .NET Core omgeving aan serverzijde (REST API) en React voor de user interface. Dit registratiesysteem bestaat uit een parking registratiesysteem, een bezoeker registratiesysteem bij de inkom en tot slot een administratie gedeelte waar men alles kan beheren.

Ik heb ervoor gekozen om de volledige front-end opnieuw te schrijven in Angular en het administratie gedeelte te koppelen aan de REST API (de andere schermen/gedeelten liet ik werken met dummy JSON data). Ik heb deze keuze gemaakt omdat ik altijd al een voorkeur heb gehad voor front-end applicaties, omdat het voor mij zeer belangrijk is dat de eindgebruiker een rustig en gebruiksvriendelijke user interface heeft. Als perfectionist vind ik er ook mijn plezier om alles esthetisch te maken en mooi op hun plaats te krijgen. Aangezien ik op dit vlak nog geen professional ben vond ik het daarom een interessante leerervaring voor mezelf. De keuze voor Angular kwam voor mij wat uit de lucht gevallen, omdat ik deze programmeertaal niet meteen aan het overwegen was. Mijn aandacht werd echter aangetrokken door Student Angular Day van NG-BE, waardoor ik meer research ben beginnen doen en zo mijn keuze na het bijwonen van hun event vastgelegd heb.

Ik heb door deze leerervaring niet alleen kennis opgedaan voor Angular maar heb tegelijkertijd ook React -een taal dat we dit semester in de lessen aanleerden- beter kunnen begrijpen en gelijkenissen/verschillen ontdekt tussen deze twee programmeertalen. Daarnaast is mijn interesse voor front-end programmeren versterkt en heb ik ook mijn voorkennis wat kunnen bijschaven. Tot slot heb ik natuurlijk ook een nieuwe, veelgebruikte programmeertaal aangeleerd die ik in de toekomst met me mee zal kunnen nemen.

GitHub repository: https://github.com/shakirahamers/Angular-Graduaatsproef

Summary

As an assignment for our graduation project, we were given the opportunity to learn a new programming language and apply it (either on the front-end or backend) to a group project we had developed for a company. In our case, we were tasked to build a visitor registration system for AllPhi, an IT consultancy firm based in Ghent, using C# in a .NET Core environment on the server-side (REST API) and React for the user interface. This registration system consists of a parking registration system, a visitor registration system at the entrance, and an administration section where everything could be managed.

I chose to rewrite the entire front-end in Angular and connect it to the REST API for the administration part (for the other parts/screens I used JSON dummy data). I made this choice because I have always had a preference for front-end applications, as I believe it is crucial for the end-user to have a clean and user-friendly interface. As a perfectionist, I also take pleasure in making everything aesthetically pleasing and well-organized. Since I am not yet a professional in this field, I found it to be an interesting learning experience for myself. The choice for Angular came as a bit of a surprise to me, as I wasn't initially considering it. However, my attention to this programming language was drawn by the Student Angular Day organized by NG-BE, which prompted me to do more research and ultimately made me determine my decision after attending their event.

Through this learning experience, I not only gained knowledge of Angular but also developed a better understanding of React, a language we had been learning in our classes this semester. I discovered similarities and differences between these two programming languages. Additionally, my interest in front-end development has been strengthened, and I was able to enhance my existing knowledge in the process. Lastly, I have learned a new widely-used programming language that I can carry with me into the future.

GitHub repository: https://github.com/shakirahamers/Angular-Graduaatsproef

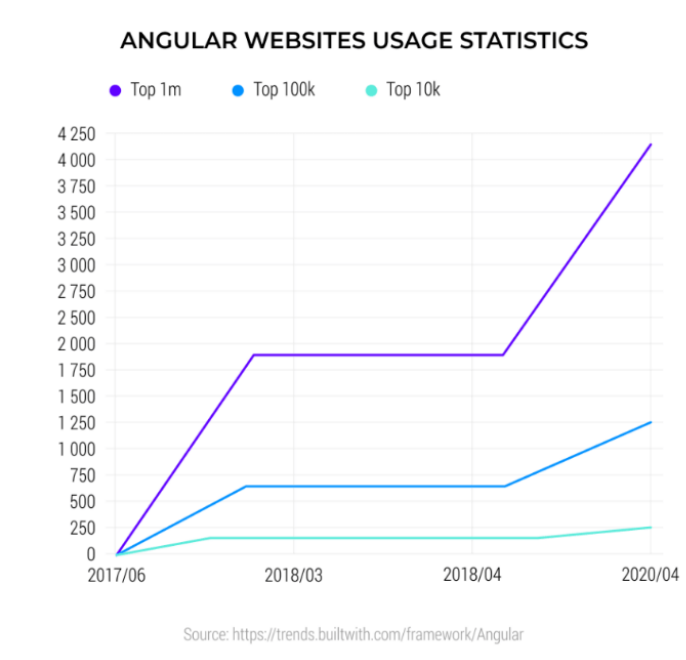
Wat is Angular?

Angular is een open-source TypeScript framework ontwikkeld door Google dat de opvolger is van AngularJs, waarvan de eerste release in 2010 was. AngularJs is een JavaScript framework waarmee dynamische webpagina's gecreëerd kunnen worden. Deze programmeertaal zorgde al voor een grote verandering in de programmeerwereld door de ontwikkeling van two-way databinding, dependency injection, event handling enzovoort.

Door het aanhoudende succes van AngularJs besloot men om een volledige herimplementatie te doen van deze programmeertaal, deze keer met TypeScript. Zo ontstond Angular 2 of eenvoudigweg Angular in 2016. Deze zorgt ervoor dat er een betere performance is voor de web ontwikkelaars en kwam uit met verschillende verbeteringen zoals hiërarchische dependency injection, component gebaseerde architectuur en volledige cross-platform (tegenover AngularJs, die enkel cross-browser is).

Men besloot daarnaast om elke zes maanden een nieuwe versie van Angular te publiceren, die krijgen de naam Angular 3, Angular 4,… Dit doen ze voor een betere stabiliteit, bugfixes, ondersteuning van nieuwe standaarden, in principe zodat Angular altijd mee is met de huidige webontwikkelingsgemeenschap. Tot de dag van vandaag (6 juni 2023) zijn er nog steeds updates, met de meest recent uitgebrachte Angular 16 op 3 mei 2023.

Zo kan men dus tot de conclusie komen dat Angular zeker aan te raden valt om aan te leren, want deze programmeertaal wordt en zal in de toekomst nog vaak gebruikt worden (zoals Microsoft Office, Samsung, Paypal).



Toelichting van mijn keuze

Inhoud van de opdracht

De opdracht die ik gemaakt heb is gekoppeld aan een groepswerk waaraan we gewerkt hebben voor een Gents consultancybedrijf: AllPhi. De opdracht voor ons was om een parking- en bezoekersregistratiesysteem te maken aan de hand van C# in een .NET Core omgeving aan serverzijde (REST API)en React voor de user interface.

De graduaatsproef houdt dan in dat we aan de hand van dit projectwerk een eigen project opstellen en dit (de front-end of de backend) koppelen met hetgeen dat we in ons projectwerk samen hebben gecreëerd.

Waarom front-end Angular?

Mijn voorkeur gaat naar front-end, omdat ik daar altijd al meer interesse had. Naast een werkende applicatie in de backend is het voor mij belangrijk dat de manier waarop deze wordt weergegeven aantrekkelijk is. De front-end is nu eenmaal hetgeen wat de eindgebruiker ziet. Als een perfectionistisch en kieskeurig persoon vind ik het ook belangrijk dat bij de front-end alles er goed uitziet en op hun plaats staat. De voldoening die ik krijg wanneer een front-end applicatie er strak en modern uitziet zorgt ervoor dat ik er zorgvuldig mijn tijd in steek om het zo gebruiksvriendelijk mogelijk te maken.

Voor ik bij mijn definitieve keuze kwam, overwoog ik eerst om de backend in Python te doen. Ik had al een tijdje een interesse in Python, maar ongelukkigerwijs was dit backend en stond het me minder aan. Maar ik kreeg tijdens de ontdekking of Python een gepaste keuze was voor mij de mogelijkheid om Angular op een interactieve manier te ontdekken, en dit zorgde ervoor dat mijn definitieve keuze gemaakt was (een diepere ingang hierop later). Tijdens het maken van deze graduaatsproef merkte ik zeker op dat de manier waarop Angular de front-end opbouwt mij enorm aanstond, en zo kreeg ik het gevoel dat ik de juiste keuze had gemaakt.

Naast mijn preferentie voor front-end was de voorkennis die ik heb ook een van de factoren waarom ik voor Angular koos. Tijdens de lessen hadden we al gebruik gemaakt van HTML, CSS en Javascript. Deze zorgden ervoor dat ik al een stevige basis had om te beginnen aangezien Angular HTML als template gebruikt, en er veel vergelijkingen zijn met React (die ook van HTML, CSS en Javascript/JSX gebruik maakt). Angular maakt ook gebruik van TypeScript. Deze programmeertaal werd soms eens tijdens lessen aangehaald, maar we zijn er nooit dieper op ingegaan. Dit heeft voor mij dan ook de interesse gewekt om TypeScript aan de hand van Angular te ontdekken.

Tot slot ben ik tot mijn besluit gekomen door Angular Student Day, georganiseerd door NG-BE op 24 maart 2023. Aan de hand van dit evenement hebben we een Angular applicatie samengesteld en heb ik zo wat basiskennis kunnen verwerven. Dit gaf me dan ook meer motivatie om te starten omdat ik wist wat ik doe, hoe ik het doe en ik mezelf kon ondersteunen aan de hand van de gecreëerde applicatie. Bij deze wil ik dan ook graag mijn appreciatie tonen aan NG-BE voor het organiseren van dit evenement.

Hoe ben ik te werk gegaan?

Telkens wanneer ik programmeer, is het voor mij belangrijk om eerst iets op mijn scherm/console te krijgen. Het maakt niet uit wat ik als output krijg, zolang het maar iets is. Want daaruit kan ik dan beginnen de correcte dingen op mijn scherm te krijgen.

Aangezien ik tijdens het projectwerk vooral heb gewerkt aan de parkingregistratie, vond ik het ook het meest toepasselijk om ook daar eerst de front-end van te maken, zodat ik daarop de basis in de vingers kon krijgen en met de verworven kennis vlotter de andere schermen kan maken (de andere schermen die ons projectwerk omvatten is een bezoeker registratie en een administratie scherm).

Ook al had ik al een redelijk afgewerkt project van Angular Student Day waarop ik me kon baseren, toch heb ik ervoor gekozen om zelf weer vanaf scratch te beginnen, alsof ik nog helemaal niks van Angular wist. Dit leek voor mij het meest evidente zodat ik op mijn eigen manier de opdracht zou maken omdat er op Angular Student Day veel ingewikkelde functies gebruikt werden, terwijl ik echt met de basis wilde beginnen en het project zou maken hoe dat ik het zelf zou doen.

De omgeving die ik gebruikt heb om de front-end applicatie te maken is Visual Studio Code, omdat we deze altijd al hebben gebruikt tijdens de lessen en omdat deze omgeving ook werd aangeraden op Angular Student Day.

Om te beginnen moest ik eerst de Angular CLI (Command Line Interface) installeren, en dit kon ik gemakkelijk doen aan de hand van een commando. Als ik dan in de terminal *ng new <projectnaam>* typ, wordt er automatisch een nieuw Angular project aangemaakt waarmee ik gemakkelijk mee van start kon gaan.

Het parking registratiesysteem heeft qua lay-out veel dezelfde schermen (zoals bijvoorbeeld bij het binnenkomen van de parking, de begroeting/afscheid). Deze schermen zijn daardoor ook simpel om op te stellen. Een zeer handige functionaliteit van Angular die ik dan ook zo ontdekt heb is dat men componenten zeer flexibel kan hergebruiken. Dit houdt in dat de lay-out exact hetzelfde is, maar de inhoud kan aangepast worden naar eigen keuze.

Op de screenshot kunt u het eerste component die ik in Angular heb aangemaakt zien. Een zeer simpele component met een titel en subtitel.

Een breakdown van deze component:

De waarde die in *selector* wordt meegegeven is de naam van de component die gebruikt wordt in de applicatie. Dit is de naam die opgeroepen wordt als men de component wilt gebruiken.

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-header',
  template: `
    <h1>{{ title }}</h1>
    <h3>{{ subtitle }}</h3>
  `,
  styles: [],
})
export class HeaderComponent {
  @Input() title: string;
  @Input() subtitle: string;

  constructor() {
    this.title = '';
    this.subtitle = '';
  }
}
```

In de *template* wordt de HTML gedefinieerd. In Angular draait alles om de HTML. Eerst wordt de HTML gebouwd en daarrond bouwt men dan al de functionaliteiten op. In dit voorbeeld heb ik ook properties gedefinieerd. Deze zijn *title* en *subtitle*.

Daaronder bevindt zich *styles*, waar de CSS meegegeven wordt. Voor overzichtelijkheid en uit gewoonte heb ik de CSS in een apart bestand neergeschreven.

Daarna komt TypeScript. Daarin gaat men de properties definiëren en in de constructor een waarde meegeven. In dit geval zullen de properties altijd een string zijn en heb ik het voor nu leeg gelaten (een lege string), omdat deze component nooit op zichzelf gebruikt zal worden, ik zal deze steeds in een ander, parent component definiëren en sowieso andere waarden meegeven.

```
<app-header [title]="'Welkom!'" [subtitle]="'Voor welk bedrijf komt u?'"></app-header>

<app-header [title]="'Bedankt!'" [subtitle]="'U kunt parkeren op plaats'"></app-header>

<app-header [title]="'Bedankt voor uw bezoek!'"
[subtitle]="'Gelieve uw nummerplaat in te voeren voor u de parking verlaat'"></app-header>

<app-header [title]="'Tot ziens!'" [subtitle]="'U mag de parking verlaten'"></app-header>
```

De volgende screenshot zijn allemaal snippets van andere componenten waar ik dit ene component heb gebruikt. Zoals er te zien valt heb ik die properties *title* en *subtitle* allemaal een andere waarde meegegeven en is de opgeroepen component de *app-header*, de naam die werd meegegeven in de *selector*.

Deze property binding hoeft niet alleen als waarde tussen een tag te zitten, je kan deze ook in de tags zelf gebruiken. Zo vond ik het dus heel handig om code te hergebruiken en efficiënt en consistent mijn HTML in componenten te declareren.

Natuurlijk bestaan mijn componenten niet alleen maar uit tags met tekst erin, ik moest ook functionerende dropdowns, buttons, inputs enzovoort maken. Mijn eerste uitdaging was een dropdown, en hier begon ik enorm veel vergelijkingen met React op te merken (waar ik later nog dieper op inga).

Nog een andere unieke functionaliteit van Angular is hun two-way databinding. In dit voorbeeld wordt bij de parkingregistratie een input aangemaakt om een nummerplaat in te voeren. Aan de hand van *ngModel* kan men de value van de input koppelen aan de property *inputValue*, die door *@Output* wordt geëmitteerd naar de parent component van deze input. Zo kan die parent component de ingevulde waarde verwerken. Aangezien het two-way databinding is, kunnen we niet alleen de waarde in deze component wijzigen, maar ook in de parent component. In deze context hoeft dat niet noodzakelijk, maar als we de geëmitteerde waarde in de parent component zouden wijzigen, zal deze wijziging ook automatisch doorgegeven worden naar deze NummerplaatComponent. Deze functionaliteit zorgt ervoor dat er minder code geschreven moet worden voor de event handling, zo is de code ook veel efficiënter.

Zo ben ik dan begonnen met het aanmaken van allemaal kleine componenten, die niet al te veel ingewikkelde functionaliteiten gebruiken. Deze componenten zijn als het ware de bouwstenen. Eenmaal ik al de bouwstenen had, was het de bedoeling om daarmee grotere, samengestelde componenten te maken (de schermen). Dit was zeer eenvoudig, en aangezien alles apart werd aangemaakt was het overzichtelijker om te weten welke componenten die ik had te kunnen hergebruiken (dit waren vooral headers of inputs die bij aankomst en verlaten van het bezoek opnieuw opgevraagd moesten worden). Het oproepen van deze componenten was ook gemakkelijk, we moeten niks importeren, aan de hand van de naam die we gegeven hebben in de selector kunnen wij doorheen de hele applicatie deze component oproepen.

Eenmaal ik die grotere componenten heb aangemaakt is het natuurlijk de bedoeling dat de routing geïmplementeerd wordt. Om deze op te stellen moeten er wel wat dingen gedaan worden maar eenmaal dit geïmplementeerd is, is het gemakkelijk deze toe te passen.

Als eerst moest ik in *app.module.ts* alles definiëren. Deze root module is een TypeScript bestand waarbij in *@NgModule* (AngularModule) ik alles moest meegeven. In *declarations* gaf ik alle componenten mee aan de hand van hun *export class* component naam, niet de *selector* naam die werd meegegeven. In *imports* gaf ik dan alle modules mee, dit zijn functionaliteiten die je kan toevoegen aan je applicatie. Hierbij is *BrowserModule* essentieel voor het opstellen van je applicatie. *AppRoutingModule* is een aparte module waar dan alle routes en paden in gedefinieerd zullen worden. In *bootstrap* wordt dan een verwijzing gedaan naar het root component (de component die geladen wordt bij het opstarten van de applicatie), hier *AppComponent*.

```
@NgModule({
  declarations: [
    AppComponent,
    BedrijfSelectComponent,
    ParkingToewijzingComponent,
    ParkingOutInvoerComponent,
    ParkingOutTotziensComponent,
    HeaderComponent,
    NummerplaatComponent,
  ],
  imports: [
    BrowserModule,
    FormsModule,
    CommonModule,
    HttpClientModule,
    AppRoutingModule,
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

App.module.ts

```
@Component({
    selector: 'app-root',
    template: `
  <router-outlet></router-outlet>
  `,
    styles: [],
})
export class AppComponent {
    title = 'graduaatsproef_angular';

}
```

App.component.ts

In deze root component wordt de hoofdstructuur van de applicatie opgesteld, maar in mijn geval gebruik ik deze als containercomponent en steek ik *router-outlet* in de template. Dit component gaat de routes renderen van de applicatie.

De router-outlet moet je nergens definiëren in een *selector* property, je moet deze aangeven in de *imports* van je component. In de *Routes* kunnen we dan de paden definiëren en het component meegeven die erbij hoort. Zo kunnen we dan doorheen de hele applicatie deze paden oproepen aan de hand van *Routerlink = "<path>"*.

```
const routes: Routes = [
  { path: '', component: BedrijfSelectComponent },
  { path: 'parkingtoewijzing', component: ParkingToewijzingComponent },
  { path: 'parkingoutinvoer', component: ParkingOutInvoerComponent },
  { path: 'parkingouttotziens', component: ParkingOutTotziensComponent },
  { path: 'header', component: HeaderComponent },
  { path: 'nummerplaat', component: NummerplaatComponent },

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

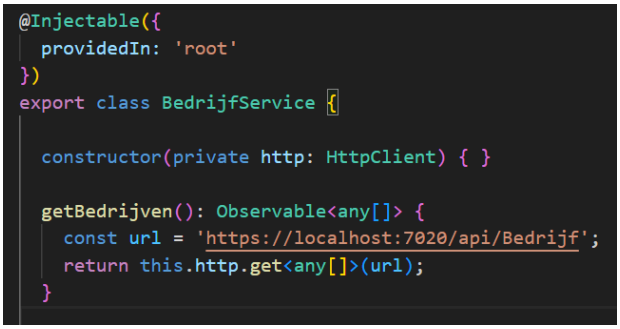
App-routing.module.ts

Aan de hand van deze basis die ik gecreëerd heb kon ik zo ook het bezoeker registratiesysteem en het admin systeem opstellen en durfde ik zo meer ingewikkeldere dingen te implementeren, zeker bij de front-end van die admin, want deze moest over verschillende functionaliteiten bezitten.

Eindresultaat - Koppeling met de backend

Eenmaal dat ik de front-end heb afgewerkt en deze werkend het gekregen met JSON dummydata, was het de bedoeling dat ik deze koppelde met de backend die we gemaakt hebben in projectwerk (een groepswerk) aan de hand van een Swagger API.

Ik heb ervoor gekozen om het administratie systeem te koppelen met de API en de rest verder met de dummy JSON data te laten werken.



De API calls opvragen in Angular ging veel vlotter en gemakkelijker dan verwacht, er moest zelfs geen Axios of dergelijke gebruikt worden zoals bij React. Door een service file aan te maken kon ik al de API calls maken door gewoon de URL mee te geven. Daarna kon ik meegeven of de URL voor een get, post, put of delete was. De *@Injectable* zorgt ervoor dat deze service gemakkelijk in alle componenten opgeroepen kan worden dus zo kon ik gemakkelijk de methodes van de service gebruiken.

De manier waarop Angular die API calls kan oproepen vind ik eenvoudiger dan ik dacht en dit zorgde er ook voor dat ik een voorkeur begon te krijgen voor deze programmeertaal. Objecten ophalen, aanmaken, updaten en verwijderen was voor mij nog nooit zo vlot geweest (zeker nadat ik een tijdje heb gevloekt op Axios in React). Door de koppeling met de backend te maken begon ik ook meer een samenhang te zien en werd ik meer tevreden over mijn project.

Mijn bevindingen

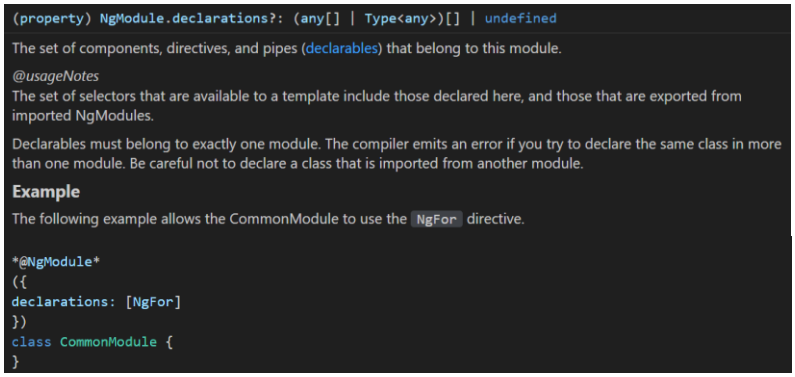
Moeilijkheden

Natuurlijk loopt het aanleren van een nieuwe programmeertaal niet altijd vlotjes en heb ik soms wel eens frustraties ervaren bij het aanmaken van dit project. Maar bij de meeste valkuilen die ik heb meegemaakt besefte ik dat de functionaliteiten eigenlijk redelijk eenvoudig in gebruik zijn, men moet gewoon begrijpen wat er exact gedaan wordt/moet worden en dan kan je deze functionaliteiten gemakkelijk hergebruiken.

Om te beginnen was het nogal moeizaam voor mij om het gebruik van TypeScript gewoon te worden. Ik denk dat dit komt omdat ik me zodanig probeerde te baseren op React en JavaScript en alleen probeerde te zoeken naar gelijkenissen en het niet zag als iets nieuws dat ik moet aanleren. Gelukkig is er voldoende documentatie van TypeScript, dus ik kon gemakkelijk mezelf informeren en de juiste toepassingen maken voor mijn project.

Daarnaast vond ik het ook moeilijk om de routing in elkaar te steken en dit komt vooral omdat ik de opmaak van modules niet echt begreep. Ik had steeds de neiging om mijn componenten in *imports* te steken in de plaats van *declarations*, en hierdoor lukte de routing natuurlijk niet. Het handige vond ik (en merkte ik pas te laat op), dat in Visual Studio Code, er veel documentatie is van de onderdelen die er in modules en componenten zijn (zeker met behulp van de “Angular Language Service” extensie). Als ik hover over bijvoorbeeld de *declarations* dan wordt dit weergegeven:

Zo wist ik dus direct wat eigenlijk waar hoort in de module, want bij *Imports* gaf men dan ook de informatie dat daarin de NgModules meegegeven worden en ik kon ook nog eens zien wat exact de geïmporteerde modules zijn.



Soms zat Angular ook zodanig eenvoudig in elkaar dat ik niet echt begreep wat ik moest doen/aan het doen was. Dit ervaarde ik ook bij het implementeren van de routing. Ik begreep niet hoe de app.component.ts wist dat *app-routing* expliciet verwees naar de routes die ik in een apart component heb gedefinieerd. Blijkbaar was dit zeer eenvoudig door het gewoon in de *imports* en *exports* te definiëren van de module. Deze aha-erlebnis ervaarde ik ook bij de componenten, waarbij *@input* en *@output* gebruikt worden om te communiceren tussen componenten.

Eenmaal ik deze connectie begreep, verstond ik veel beter hoe ik routing moest implementeren bij mijn andere systemen (aangezien ik eerst begonnen was met het parkingregistratie systeem) en ging het dus ook veel vlotter om de basis van de applicaties te implementeren.

Hetgeen waar ik dan wat mee gesukkeld had tijdens de koppeling met de backend, was de validatie. Hierbij heb ik gebruik gemaakt van een Form waarin ik dan zo de validators in kon steken, zodat ik zo het gehele formulier kon valideren. Het geheel is een *FormGroup*, en alle aparte onderdelen (zoals bijvoorbeeld de naam van een bedrijf, hun e-mail, telefoonnummer) zijn *FormControls*, daarin steek ik dan de waarde in. Hierdoor moest ik jammer genoeg wel de two-way databinding met *NgModel* laten vallen, want deze gebruikte ik voor ik de validatie toevoegde. Beide kunnen eigenlijk wel samen gebruikt worden, maar dit werd niet aangeraden aangezien het gebruik van zowel *FormControl* als *NgModel* samen voor conflicten kan zorgen.

Vergelijking met React

Er zijn veel gelijkenissen maar ook verschillen tussen Angular en React. De grootste verschillen die ik zelf ervaarde tijdens het maken van dit project was de property binding, de two-way databinding en de Model View Controller (MVC).

Zoals ik eerder vermeldde zorgde die property binding ervoor dat ik heel gemakkelijk componenten kon hergebruiken. Dit zorgt niet enkel voor efficiëntie en vermindering van de code, maar zorgt er ook voor dat mijn applicatie consistenter is, omdat dezelfde styling gebruikt wordt. Daarnaast moest ik ook veel minder sukkelen met de styling (om bijvoorbeeld iets op de correcte plaats op het scherm te krijgen), want als het met 1 component gelukt is hergebruik ik die gewoon maar pas ik de waarde aan zodat het past bij mijn parent-component. Natuurlijk is het hergebruik van componenten ook van toepassing bij React, maar door de property binding was dit veel uitgebreider.

Het handige van two-way databinding is dat er in Angular niet iets als een UseState hook in React gebruikt moet worden. De waarde wordt gewoon automatisch geüpdatet en je kan alles oproepen aan de hand van de *[(NgModel)]*. Ik moest me dan geen zorgen maken dat ik een ingevoerde waarde vergat up te daten of dergelijke en ik kon mij zo beter focussen op de andere moeilijkheden van Angular. Deze functionaliteit heb ik vaak gebruikt in mijn project en hierdoor gaat mijn voorkeur ook wel meer naar Angular, omdat deze programmeertaal veelgebruikte functionaliteiten vereenvoudigt.

De Model View Controller heb ik niet expliciet gebruikt in mijn Angular project, omdat er niet extreem veel code wordt gecreëerd waardoor de componenten onleesbaar beginnen te worden. Maar deze functionaliteit vond ik ook zeer interessant om over te leren, omdat je de code zeer gemakkelijk kan splitsen in de Model (de logica in verband met de functionaliteit),

View (de presentatie van de gegevens op de user interface, de HTML dus) en Controller (de tussenpersoon tussen Model en View). Deze splitsing werd gebruikt in het project dat we gemaakt hadden op Angular Student Day, waar Angular nog volledig nieuw was voor mij, en zo begreep ik veel gemakkelijk wat alles exact inhoudt. Zeker aangezien dat project veel ingewikkelder en groter in omvang was in vergelijking met het project dat ik had aangemaakt (wij moesten toen verder bouwen op al grotendeels aangemaakte code en zo functionaliteiten toevoegen).

Tot slot is er natuurlijk ook het verschil met het gebruik van TypeScript in Angular en JavaScript in React. Hiervoor gaat mijn keuze toch meer naar JavaScript, omdat deze gemakkelijker in gebruik is. TypeScript heeft veel verschillende functionaliteiten die ik niet altijd begrijp/onthoud, maar gelukkig wel gemakkelijk op te zoeken zijn. Ik zie zeker vergelijking met C# en kon het zo wel langzamerhand beginnen begrijpen.

Conclusie: wat leerde ik hieruit?

Volgens mij is de meest opvallende bevinding die ik heb opgemerkt dat Angular zeker geen extreem moeilijke programmeertaal is om (op jezelf) aan te leren, maar er zeker wel kennis van HTML, CSS en JavaScript moet zijn (en eventueel ook C# om bepaalde functionaliteiten van TypeScript gemakkelijker te kunnen herkennen). Daarom vond ik het ook een goede keuze dat ik Angular koos voor mijn graduaatsproef, omdat ik in hetzelfde semester ook React aangeleerd kreeg en zo stapsgewijs gelijkenissen en verschillen kon opsporen en zo niet alleen Angular, maar ook React beter kon begrijpen.

Zoals ook eerder aangehaald is in Angular eigenlijk HTML de template waarop men functionaliteiten bouwt, en zo heb ik ook meer moeite gestoken in de HTML en CSS van deze applicatie te perfectioneren en uit te breiden.

Aangezien mijn voorkeur naar front-end gaat en ik in de toekomst ook liever daarop zou focussen, ben ik er vrij zeker van dat ik Angular nog zal gebruiken, omdat deze programmeertaal enorm populair is en volgens mij ook een tijdje zal blijven.

Daarnaast was ik ook aan het experimenteren welke CSS framework mij het meest aanstond, want in de lessen hebben we zowel Tailwind, Bootstrap als MateralUI gezien. In het groepswerk in projectwerk was ik begonnen met het experimenteren van zowel Bootstrap als MaterialUI dus daarom besloot ik om mijn graduaatsproef in Tailwind te doen. Hiermee ontdekte ik dat Tailwind zeker mijn voorkeur heeft, deze bevat enorm veel verschillende styling functionaliteiten (veel uitgebreider dan BootStrap) die zeer gemakkelijk te implementeren zijn en bevat ook zeer goede documentatie die ik van de Tailwind website kon overnemen en toepassen.

Om dus tot een besluit te komen heeft deze graduaatsproef ervoor gezorgd dat ik niet alleen mijn kennis kon uitbreiden en bijschaven, maar ook kon ontdekken wat mij het meest bevalt om (in de toekomst) te gebruiken bij het programmeren van applicaties.

Bronnenlijst

GOOGLE. (2010-2023). Property Binding. Van https://angular.io/guide/property-binding.

GOOGLE. (2010-2023). Two-way Binding. Van https://angular.io/guide/two-way-binding.

LAMBREGHTS Maxime. (16 april 2016). Wat is Angular?. Van https://master-it.nl/blog/wat-is-angular/.

W3SCHOOLS. (1999-2023). AngularJS Introduction. Van https://www.w3schools.com/angular/angular_intro.asp.

HARTMAN James. (22 april 2023). Angular Version List & History – Angular 2,4,5,6,7,8. Van https://www.guru99.com/angularjs-1-vs-2-vs-4-vs-5-difference.html#:~:text=After%20releasing%20Angular%20JS%2C%20the,with%20other%20improvements%20and%20tweaks.

2MUCHCOFFEE. (29 juli 2020). 15 Surprising Stats About Angular. Van https://dev.to/2muchcoffeecom/15-surprising-stats-about-angular-3fh7.